

Final Assignment

June 30, 2025

Extracting and Visualizing Stock Data

Description

Extracting essential data from a dataset and displaying it is a necessary part of data science; therefore individuals can make correct decisions based on the data. In this assignment, you will extract some stock data, you will then display this data in a graph.

Table of Contents

- Define a Function that Makes a Graph
- Question 1: Use yfinance to Extract Stock Data
- Question 2: Use Webscraping to Extract Tesla Revenue Data
- Question 3: Use yfinance to Extract Stock Data
- Question 4: Use Webscraping to Extract GME Revenue Data
- Question 5: Plot Tesla Stock Graph
- Question 6: Plot GameStop Stock Graph

Estimated Time Needed: 30 min

Note:- If you are working Locally using anaconda, please uncomment the following code and execute it. Use the version as per your python version.

```
[1]: !pip install yfinance
      !pip install bs4
      !pip install nbformat
      !pip install --upgrade plotly
```

Requirement already satisfied: yfinance in /opt/conda/lib/python3.12/site-packages (0.2.64)

Requirement already satisfied: pandas>=1.3.0 in /opt/conda/lib/python3.12/site-packages (from yfinance) (2.3.0)

Requirement already satisfied: numpy>=1.16.5 in /opt/conda/lib/python3.12/site-packages (from yfinance) (2.3.1)

Requirement already satisfied: requests>=2.31 in /opt/conda/lib/python3.12/site-packages (from yfinance) (2.32.3)

Requirement already satisfied: multitasking>=0.0.7 in /opt/conda/lib/python3.12/site-packages (from yfinance) (0.0.11)

Requirement already satisfied: platformdirs>=2.0.0 in /opt/conda/lib/python3.12/site-packages (from yfinance) (4.3.6)

Requirement already satisfied: pytz>=2022.5 in /opt/conda/lib/python3.12/site-packages (from yfinance) (2024.2)

Requirement already satisfied: frozendict>=2.3.4 in /opt/conda/lib/python3.12/site-packages (from yfinance) (2.4.6)

Requirement already satisfied: peewee>=3.16.2 in /opt/conda/lib/python3.12/site-packages (from yfinance) (3.18.1)

Requirement already satisfied: beautifulsoup4>=4.11.1 in /opt/conda/lib/python3.12/site-packages (from yfinance) (4.13.4)

Requirement already satisfied: curl_cffi>=0.7 in /opt/conda/lib/python3.12/site-packages (from yfinance) (0.11.4)

Requirement already satisfied: protobuf>=3.19.0 in /opt/conda/lib/python3.12/site-packages (from yfinance) (6.31.1)

Requirement already satisfied: websockets>=13.0 in /opt/conda/lib/python3.12/site-packages (from yfinance) (15.0.1)

Requirement already satisfied: soupsieve>1.2 in /opt/conda/lib/python3.12/site-packages (from beautifulsoup4>=4.11.1->yfinance) (2.5)

Requirement already satisfied: typing-extensions>=4.0.0 in /opt/conda/lib/python3.12/site-packages (from beautifulsoup4>=4.11.1->yfinance) (4.12.2)

Requirement already satisfied: cffi>=1.12.0 in /opt/conda/lib/python3.12/site-packages (from curl_cffi>=0.7->yfinance) (1.17.1)

Requirement already satisfied: certifi>=2024.2.2 in /opt/conda/lib/python3.12/site-packages (from curl_cffi>=0.7->yfinance) (2024.12.14)

Requirement already satisfied: python-dateutil>=2.8.2 in /opt/conda/lib/python3.12/site-packages (from pandas>=1.3.0->yfinance) (2.9.0.post0)

Requirement already satisfied: tzdata>=2022.7 in /opt/conda/lib/python3.12/site-packages (from pandas>=1.3.0->yfinance) (2025.2)

Requirement already satisfied: charset_normalizer<4,>=2 in /opt/conda/lib/python3.12/site-packages (from requests>=2.31->yfinance) (3.4.1)

Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.12/site-packages (from requests>=2.31->yfinance) (3.10)

Requirement already satisfied: urllib3<3,>=1.21.1 in /opt/conda/lib/python3.12/site-packages (from requests>=2.31->yfinance) (2.3.0)

Requirement already satisfied: pycparser in /opt/conda/lib/python3.12/site-packages (from cffi>=1.12.0->curl_cffi>=0.7->yfinance) (2.22)

Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.12/site-packages (from python-dateutil>=2.8.2->pandas>=1.3.0->yfinance) (1.17.0)

Requirement already satisfied: bs4 in /opt/conda/lib/python3.12/site-packages (0.0.2)

Requirement already satisfied: beautifulsoup4 in /opt/conda/lib/python3.12/site-packages (from bs4) (4.13.4)

Requirement already satisfied: soupsieve>1.2 in /opt/conda/lib/python3.12/site-packages (from beautifulsoup4->bs4) (2.5)

Requirement already satisfied: typing-extensions>=4.0.0 in /opt/conda/lib/python3.12/site-packages (from beautifulsoup4->bs4) (4.12.2)

Requirement already satisfied: nbformat in /opt/conda/lib/python3.12/site-

```

packages (5.10.4)
Requirement already satisfied: fastjsonschema>=2.15 in
/opt/conda/lib/python3.12/site-packages (from nbformat) (2.21.1)
Requirement already satisfied: jsonschema>=2.6 in
/opt/conda/lib/python3.12/site-packages (from nbformat) (4.23.0)
Requirement already satisfied: jupyter-core!=5.0.*,>=4.12 in
/opt/conda/lib/python3.12/site-packages (from nbformat) (5.7.2)
Requirement already satisfied: traitlets>=5.1 in /opt/conda/lib/python3.12/site-
packages (from nbformat) (5.14.3)
Requirement already satisfied: attrs>=22.2.0 in /opt/conda/lib/python3.12/site-
packages (from jsonschema>=2.6->nbformat) (25.1.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in
/opt/conda/lib/python3.12/site-packages (from jsonschema>=2.6->nbformat)
(2024.10.1)
Requirement already satisfied: referencing>=0.28.4 in
/opt/conda/lib/python3.12/site-packages (from jsonschema>=2.6->nbformat)
(0.36.2)
Requirement already satisfied: rpds-py>=0.7.1 in /opt/conda/lib/python3.12/site-
packages (from jsonschema>=2.6->nbformat) (0.22.3)
Requirement already satisfied: platformdirs>=2.5 in
/opt/conda/lib/python3.12/site-packages (from jupyter-
core!=5.0.*,>=4.12->nbformat) (4.3.6)
Requirement already satisfied: typing-extensions>=4.4.0 in
/opt/conda/lib/python3.12/site-packages (from
referencing>=0.28.4->jsonschema>=2.6->nbformat) (4.12.2)
Requirement already satisfied: plotly in /opt/conda/lib/python3.12/site-packages
(5.24.1)
Collecting plotly
  Downloading plotly-6.2.0-py3-none-any.whl.metadata (8.5 kB)
Collecting narwhals>=1.15.1 (from plotly)
  Downloading narwhals-1.44.0-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: packaging in /opt/conda/lib/python3.12/site-
packages (from plotly) (24.2)
Downloading plotly-6.2.0-py3-none-any.whl (9.6 MB)
      9.6/9.6 MB
152.4 MB/s eta 0:00:00
Downloading narwhals-1.44.0-py3-none-any.whl (365 kB)
Installing collected packages: narwhals, plotly
  Attempting uninstall: plotly
    Found existing installation: plotly 5.24.1
    Uninstalling plotly-5.24.1:
      Successfully uninstalled plotly-5.24.1
Successfully installed narwhals-1.44.0 plotly-6.2.0

```

```

[2]: import yfinance as yf
import pandas as pd
import requests

```

```
from bs4 import BeautifulSoup
import plotly.graph_objects as go
from plotly.subplots import make_subplots
```

```
[3]: import plotly.io as pio
pio.renderers.default = "iframe"
```

In Python, you can ignore warnings using the warnings module. You can use the filterwarnings function to filter or ignore specific warning messages or categories.

```
[4]: import warnings
# Ignore all warnings
warnings.filterwarnings("ignore", category=FutureWarning)
```

0.1 Define Graphing Function

In this section, we define the function `make_graph`. You don't have to know how the function works, you should only care about the inputs. It takes a dataframe with stock data (dataframe must contain Date and Close columns), a dataframe with revenue data (dataframe must contain Date and Revenue columns), and the name of the stock.

```
[5]: def make_graph(stock_data, revenue_data, stock):
    fig = make_subplots(rows=2, cols=1, shared_xaxes=True,
        ↳ subplot_titles=("Historical Share Price", "Historical Revenue"),
        ↳ vertical_spacing = .3)
    stock_data_specific = stock_data[stock_data.Date <= '2021-06-14']
    revenue_data_specific = revenue_data[revenue_data.Date <= '2021-04-30']
    fig.add_trace(go.Scatter(x=pd.to_datetime(stock_data_specific.Date,
        ↳ infer_datetime_format=True), y=stock_data_specific.Close.astype("float"),
        ↳ name="Share Price"), row=1, col=1)
    fig.add_trace(go.Scatter(x=pd.to_datetime(revenue_data_specific.Date,
        ↳ infer_datetime_format=True), y=revenue_data_specific.Revenue.
        ↳ astype("float"), name="Revenue"), row=2, col=1)
    fig.update_xaxes(title_text="Date", row=1, col=1)
    fig.update_xaxes(title_text="Date", row=2, col=1)
    fig.update_yaxes(title_text="Price ($US)", row=1, col=1)
    fig.update_yaxes(title_text="Revenue ($US Millions)", row=2, col=1)
    fig.update_layout(showlegend=False,
        height=900,
        title=stock,
        xaxis_rangeflider_visible=True)
    fig.show()
    from IPython.display import display, HTML
    fig_html = fig.to_html()
    display(HTML(fig_html))
```

Use the `make_graph` function that we've already defined. You'll need to invoke it in questions 5 and 6 to display the graphs and create the dashboard. > **Note: You don't need to redefine**

the function for plotting graphs anywhere else in this notebook; just use the existing function.

0.2 Question 1: Use yfinance to Extract Stock Data

Using the `Ticker` function enter the ticker symbol of the stock we want to extract data on to create a ticker object. The stock is Tesla and its ticker symbol is TSLA.

```
[6]: tesla = yf.Ticker("TSLA")
tesla
```

```
[6]: yfinance.Ticker object <TSLA>
```

Using the ticker object and the function `history` extract stock information and save it in a dataframe named `tesla_data`. Set the `period` parameter to "max" so we get information for the maximum amount of time.

```
[8]: tesla_data = tesla.history(period= "max")
```

Reset the index using the `reset_index(inplace=True)` function on the `tesla_data` DataFrame and display the first five rows of the `tesla_data` dataframe using the `head` function. Take a screenshot of the results and code from the beginning of Question 1 to the results below.

```
[10]: tesla_data.reset_index(inplace=True)
```

```
[11]: tesla_data.head(5)
```

```
[11]:
```

| | Date | Open | High | Low | Close | \ |
|---|---------------------------|----------|----------|----------|----------|---|
| 0 | 2010-06-29 00:00:00-04:00 | 1.266667 | 1.666667 | 1.169333 | 1.592667 | |
| 1 | 2010-06-30 00:00:00-04:00 | 1.719333 | 2.028000 | 1.553333 | 1.588667 | |
| 2 | 2010-07-01 00:00:00-04:00 | 1.666667 | 1.728000 | 1.351333 | 1.464000 | |
| 3 | 2010-07-02 00:00:00-04:00 | 1.533333 | 1.540000 | 1.247333 | 1.280000 | |
| 4 | 2010-07-06 00:00:00-04:00 | 1.333333 | 1.333333 | 1.055333 | 1.074000 | |

| | Volume | Dividends | Stock Splits |
|---|-----------|-----------|--------------|
| 0 | 281494500 | 0.0 | 0.0 |
| 1 | 257806500 | 0.0 | 0.0 |
| 2 | 123282000 | 0.0 | 0.0 |
| 3 | 77097000 | 0.0 | 0.0 |
| 4 | 103003500 | 0.0 | 0.0 |

0.3 Question 2: Use Web scraping to Extract Tesla Revenue Data

Use the `requests` library to download the webpage <https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/revenue.htm> Save the text of the response as a variable named `html_data`.

```
[22]: url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
↳ IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/revenue.htm"
html_data = requests.get(url).text
```

Parse the html data using beautiful_soup using parser i.e html5lib or html.parser.

```
[24]: soup = BeautifulSoup(html_data, "html.parser")
```

Using BeautifulSoup or the read_html function extract the table with Tesla Revenue and store it into a dataframe named tesla_revenue. The dataframe should have columns Date and Revenue.

Step-by-step instructions

Here are the step-by-step instructions:

1. Create an Empty DataFrame
2. Find the Relevant Table
3. Check for the Tesla Quarterly Revenue Table
4. Iterate Through Rows in the Table Body
5. Extract Data from Columns
6. Append Data to the DataFrame

[Click here](#) if you need help locating the table

Below is the code to isolate the table, you will now need to loop through the rows and columns

```
soup.find_all("tbody")[1]
```

If you want to use the read_html function the table is located at index 1

We are focusing on quarterly revenue in the lab.

```
[85]: tesla_revenue = pd.read_html(html_data)[1]
tesla_revenue.columns = ["Date", "Revenue"]
```

Execute the following line to remove the comma and dollar sign from the Revenue column.

```
[86]: tesla_revenue["Revenue"] = tesla_revenue['Revenue'].str.
↳ replace(',', '\$', "", regex=True)
```

Execute the following lines to remove an null or empty strings in the Revenue column.

```
[88]: tesla_revenue.dropna(inplace=True)

tesla_revenue = tesla_revenue[tesla_revenue['Revenue'] != ""]
```

Display the last 5 row of the tesla_revenue dataframe using the tail function. Take a screenshot of the results.

```
[89]: tesla_revenue.tail(5)
```

```
[89]:
```

| | Date | Revenue |
|----|------------|---------|
| 48 | 2010-09-30 | 31 |
| 49 | 2010-06-30 | 28 |
| 50 | 2010-03-31 | 21 |
| 52 | 2009-09-30 | 46 |
| 53 | 2009-06-30 | 27 |

0.4 Question 3: Use yfinance to Extract Stock Data

Using the `Ticker` function enter the ticker symbol of the stock we want to extract data on to create a ticker object. The stock is GameStop and its ticker symbol is `GME`.

```
[101]: gamestop = yf.Ticker("GME")
```

Using the ticker object and the function `history` extract stock information and save it in a dataframe named `gme_data`. Set the `period` parameter to `"max"` so we get information for the maximum amount of time.

```
[102]: gme_data = gamestop.history(period="max")
```

Reset the index using the `reset_index(inplace=True)` function on the `gme_data` DataFrame and display the first five rows of the `gme_data` dataframe using the `head` function. Take a screenshot of the results and code from the beginning of Question 3 to the results below.

```
[103]: gme_data.reset_index(inplace=True)
gme_data.head(5)
```

```
[103]:
```

| | Date | Open | High | Low | Close | Volume | \ |
|---|---------------------------|----------|----------|----------|----------|----------|---|
| 0 | 2002-02-13 00:00:00-05:00 | 1.620128 | 1.693350 | 1.603296 | 1.691667 | 76216000 | |
| 1 | 2002-02-14 00:00:00-05:00 | 1.712707 | 1.716073 | 1.670626 | 1.683250 | 11021600 | |
| 2 | 2002-02-15 00:00:00-05:00 | 1.683251 | 1.687459 | 1.658002 | 1.674834 | 8389600 | |
| 3 | 2002-02-19 00:00:00-05:00 | 1.666418 | 1.666418 | 1.578047 | 1.607504 | 7410400 | |
| 4 | 2002-02-20 00:00:00-05:00 | 1.615920 | 1.662210 | 1.603296 | 1.662210 | 6892800 | |

| | Dividends | Stock Splits |
|---|-----------|--------------|
| 0 | 0.0 | 0.0 |
| 1 | 0.0 | 0.0 |
| 2 | 0.0 | 0.0 |
| 3 | 0.0 | 0.0 |
| 4 | 0.0 | 0.0 |

0.5 Question 4: Use Webscraping to Extract GME Revenue Data

Use the `requests` library to download the webpage <https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html>. Save the text of the response as a variable named `html_data_2`.

```
[107]: url_2 = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
↳IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html"

html_data_2 = requests.get(url_2).text
```

Parse the html data using beautiful_soup using parser i.e html5lib or html.parser.

```
[110]: soup_2 = BeautifulSoup(html_data_2,"html.parser")
```

Using BeautifulSoup or the read_html function extract the table with GameStop Revenue and store it into a dataframe named gme_revenue. The dataframe should have columns Date and Revenue. Make sure the comma and dollar sign is removed from the Revenue column.

Note: Use the method similar to what you did in question 2.

Click here if you need help locating the table

Below is the code to isolate the table, you will now need to loop through the rows and columns

```
soup.find_all("tbody")[1]
```

If you want to use the read_html function the table is located at index 1

```
[120]: gme_revenue = pd.read_html(html_data_2)[1]
gme_revenue.columns = ["Date", "Revenue"]
gme_revenue["Revenue"] = gme_revenue["Revenue"].
↳replace(r"^[A-Za-z0-9\s]", "", regex=True)
```

Display the last five rows of the gme_revenue dataframe using the tail function. Take a screenshot of the results.

```
[121]: gme_revenue.tail(5)
```

```
[121]:
```

| | Date | Revenue |
|----|------------|---------|
| 57 | 2006-01-31 | 1667 |
| 58 | 2005-10-31 | 534 |
| 59 | 2005-07-31 | 416 |
| 60 | 2005-04-30 | 475 |
| 61 | 2005-01-31 | 709 |

0.6 Question 5: Plot Tesla Stock Graph

Use the make_graph function to graph the Tesla Stock Data, also provide a title for the graph. Note the graph will only show data upto June 2021.

Hint

You just need to invoke the make_graph function with the required parameter to print the graph.


```
[130]: make_graph(tesla_data,tesla_revenue,'Tesla')
```

```
/tmp/ipykernel_2092/109047474.py:5: UserWarning:
```

The argument 'infer_datetime_format' is deprecated and will be removed in a future version. A strict version of it is now the default, see <https://pandas.pydata.org/pdeps/0004-consistent-to-datetime-parsing.html>. You can safely remove this argument.

```
/tmp/ipykernel_2092/109047474.py:6: UserWarning:
```

The argument 'infer_datetime_format' is deprecated and will be removed in a future version. A strict version of it is now the default, see <https://pandas.pydata.org/pdeps/0004-consistent-to-datetime-parsing.html>. You can safely remove this argument.

<IPython.core.display.HTML object>

0.7 Question 6: Plot GameStop Stock Graph

Use the `make_graph` function to graph the GameStop Stock Data, also provide a title for the graph. The structure to call the `make_graph` function is `make_graph(gme_data, gme_revenue, 'GameStop')`. Note the graph will only show data upto June 2021.

Hint

You just need to invoke the `make_graph` function with the required parameter to print the graph.

```
[131]: make_graph(gme_data,gme_revenue,"GameStop")
```

```
/tmp/ipykernel_2092/109047474.py:5: UserWarning:
```

The argument 'infer_datetime_format' is deprecated and will be removed in a future version. A strict version of it is now the default, see <https://pandas.pydata.org/pdeps/0004-consistent-to-datetime-parsing.html>. You can safely remove this argument.

```
/tmp/ipykernel_2092/109047474.py:6: UserWarning:
```

The argument 'infer_datetime_format' is deprecated and will be removed in a future version. A strict version of it is now the default, see <https://pandas.pydata.org/pdeps/0004-consistent-to-datetime-parsing.html>. You can safely remove this argument.

<IPython.core.display.HTML object>

About the Authors:

Joseph Santarcangelo has a PhD in Electrical Engineering, his research focused on using machine learning, signal processing, and computer vision to determine how videos impact human cognition. Joseph has been working for IBM since he completed his PhD.

Azim Hirjani

0.8 Change Log

| Date (YYYY-MM-DD) | Version | Changed By | Change Description |
|-------------------|---------|---------------|-----------------------------|
| 2022-02-28 | 1.2 | Lakshmi Holla | Changed the URL of GameStop |
| 2020-11-10 | 1.1 | Malika Singla | Deleted the Optional part |
| 2020-08-27 | 1.0 | Malika Singla | Added lab to GitLab |

##

© IBM Corporation 2020. All rights reserved.